Università di Pisa
Dipartimento di Informatica

Technical Report: TR-08-25

Supporting user–friendly design of flexible Business Processes in StPowla

P. Fantini
MIP - School of Management
Politecnico di Milano

C. Montangero
Dipartimento di Informatica
Università di Pisa

C. Palasciano
MIP - School of Management
Politecnico di Milano

S. Reiff-Marganiec
Department of Computer Science
University of Leicester

L. Semini
Dipartimento di Informatica
Università di Pisa

September 24, 2008

Address: Largo B. Pontecorvo 3, 56127 Pisa, Italy. Tel: +39 050 2212700 Fax: +39 050 2212726

Supporting user–friendly design of flexible Business Processes in StPowla

P. Fantini
MIP - School of Management
Politecnico di Milano

C. Montangero
Dipartimento di Informatica
Università di Pisa

C. Palasciano
MIP - School of Management
Politecnico di Milano

S. Reiff-Marganiec
Department of Computer Science
University of Leicester

L. Semini
Dipartimento di Informatica
Università di Pisa

September 24, 2008

Abstract

The integration of BPM and SOA has the potential to lead to increased agility, lower development and maintenance costs and a better alignment between business and IT. However, it still requires large efforts by highly skilled personnel. The Service-Targeted Policy-Oriented WorkfLow Approach attacks this problem by integrating workflows and services with *policies* to clearly distinguish between the *core* process description and the *variations*. This paper presents a first assessment of the impact of the approach on the design of business processes. To this end, we exploit a case study, namely the activation of VoIP services by a re–seller of telecommunication services.

1 Introduction

We are facing an unprecedented degree and speed of change in the requirements for software systems, since software lives nowadays in a changing open world [4]. For instance, to reduce time-to-market, enterprises federate their operations by networking via Web services, and these federations can change to follow evolving business goals. On a smaller scale, processes may need to adapt to temporary shortage of resources by simplifying, or even skip, some steps. Engineers need to cope with these environmental changes while the supporting

software system is operating. Traditionally, domains where new features or fixes had to be incorporated into a running software were rare, a notable exception being telecommunications. Evolution is becoming more and more important in many other domains, notably in business processes: software evolution, traditionally practiced as an off–line activity, nowadays must be accommodated at run–time. The integration of Business Process Management (BPM) and Service Oriented Architecture (SOA) has been recognized as a promising approach in this respect [12].

However, the integration of BPM and SOA still requires large efforts by highly skilled personnel. Currently, the business rules introduced by business roles like sales or technical managers need to be mediated by a business analyst who, thanks to his knowledge of the business processes, transforms them into directives to the programmers for updating the workflows, e.g. in BPEL.

This paper addresses the problem of designing business processes to provide the stakeholders with *natural* ways (i) to understand the basic structure of the workflows, and (ii) to adapt the workflow behaviour in response to business needs. We leverage the Service-Targeted Policy-Oriented WorkfLow Approach (StPowla – to be read like "Saint Paula"), an approach to business process modelling that merges three ingredients: a workflow notation, a policy language, and the SOA [8, 5].

A policy based approach clearly distinguishes between a *core* description of the process and its *variations*, which can be specified by declarative rules, and can be deployed and removed dynamically. This foster Business Process flexibility, by raising the abstraction level at which variations are specified, at the same time providing a suitable implementation technique. Indeed, policies have been successfully applied to deal with change and variability in several software domains, e.g. software for telecommunications.

In the approach, business tasks are ultimately carried out by *services*, i.e. computational entities that are characterized by two series of parameters: the *invocation* parameters (related to their functionalities), and the *Service Level* (SL) parameters, related to the resources they exploit to carry out their job: Stakeholders can adapt the core workflows by requiring higher or lower service levels, therefore consuming more or less resources.

The combination of workflows, SOA, and policies can be exploited at its best, if a coherent design strategy is adopted to foster flexibility. In a nutshell, such a strategy is to find the best balance between (i) keeping the workflows simple, i.e., without explicit choices that depend on the quantity/quality of resources available to the tasks, and (ii) providing large and foreseeing ranges of choices to the policies, to support modelling the business rules as they pop up.

The kind and granularity of the 'resources' that are identified in the business domain define a significant part of the design space in StPowla, the rest being related to task functionalities. Since StPowla addresses the integration of business processes and SOA at a high level of abstraction, close to the business goals, the kind of resources associated to service levels are often more abstract than those usually addressed when establishing service level agreements, like bandwidth, power, etc. For instance, a task of a given type may need higher levels of authorization in given circumstances, and lower levels in others. In StPowla, the authorizing business roles are seen as resources, ordered along

an `AuthorizationLevel` dimension: the identification of these dimensions is a key design activity in StPowla .

It is worthwhile to point out that the ideas presented here result from the cooperation of authors with different backgrounds, namely software engineering and business management. We think that this cooperation is key to following realistic paths in our investigations.

The next section elaborates the problem we want to face, via a case study. Section 3 reviews StPowla, and Section 4 revisits the case study. We discuss our approach in section 5 and related work in section 6. Finally, we wrap up with conclusions and future work.

2 The problem, in an example

To illustrate our point, we exploit a case study from the telecommunication sector, proposed in the SENSORIA project [25]. In this process, dubbed *VoIP service activation*, a re–seller activates a VoIP service on wireless devices that support WiMAX and WiFi connections. The customers may be either individual or business customers, with different requirements on the provided service and procuring process.

The case study centers on the activation of a VoIP service by a telecommunication (Telco) operator for individual or business customers. The Telco operator, in its turn, buys telephony services from certified TIER 1 international suppliers.

The workflow consists of four tasks in sequence:

1:Customer request – the customer connects to the Telco Customer Relationship Management application (CRM), identifies itself and chooses the requested service features and Quality of Service (QoS).

2:Service Offer – the CRM elaborates detailed customer request data plus the relevant technical reports and forwards the result to the Sales Department (supported by an Order Management application). The Sales Department chooses the third–party supplier that best matches the requested QoS while maximizing the company profits. Finally, a service offer including service features, validated QoS, fees and terms is sent to the customer.

3:PreDelivery – once the customer agrees to the offer, the Technical Department is advised to perform service tests to validate the QoS offered. Then, a formal sales contract is issued for the customer, with legal consultancy from the Internal Legal Department, if necessary.

4:Activation – after the customer agrees upon the service contract, the service is installed and the relevant information is added into the Billing Platform application.

Consider now the following scenario: The Sales Department is handling a very big order from a major customer and the Technical Department is overloaded. Since 80% of the testing activities are for less relevant, individual customers, the

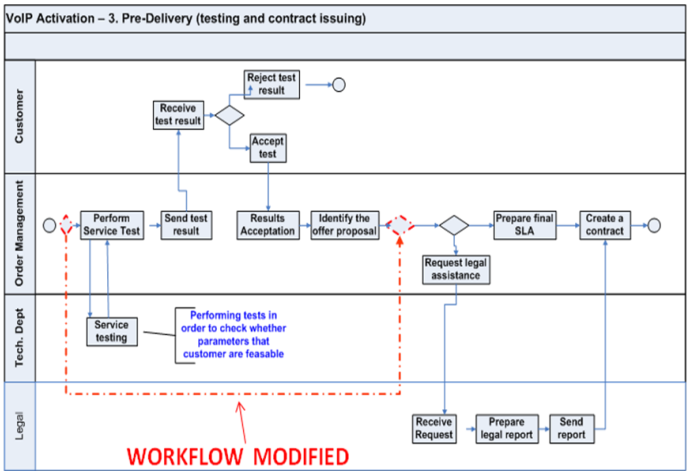

Figure 1: Required changes to the VoIP workflow

Sales Manager envisages to spare on the related resources. Since this stakeholder has the necessary authority, he issues a new business rule (we call it R1):

```
R1:  Until the end of year, no testing must be performed
for orders of small amount from individual customers.
```

In order to comply to the new business rule, the business analyst (BA) has to change all the workflows that include testing activities provided by the Technical Department, so to reduce its workload. Here is a likely scenario:

Step 1: Search. The BA has to look carefully into all the workflow diagrams to find testing activities. In particular, the VoIP workflow is divided into four diagrams, one for each task. Just to show the complexity of these diagrams, Figure 1 presents the most interesting for our purposes (Predelivery).

Step 2: Found. As per the previous description of the tasks, testing activities are found in Task 3, Pre-Delivery.

Step 3: Analyze/Change. Accordingly, the BA has to analyze the workflow of Task 3 in detail, and change it according to rule R1, as shown by the dotted part of Figure 1.

Step 4: Continue. The previous activities must be repeated for each of the other business processes of the company.

Step 5: Record. The BA has to release a new version of the workflow and record in the BA files the rule R1 and the relevant changes made to the workflows. In fact, rule R1 has lost its identity in the new workflows.

Step 6: Hand to IT department. The BA asks the IT manager to change the program code accordingly. In business practice, often business process workflow representations and program code are not connected with automatic transformation tools. In this cases, the BA has to hand the to-be-modified workflows over to the IT manager, to have the changes implemented in the code. The new

programs must then be tested and finally deployed in the business environment. The BA has to follow up the new code development and implementation until finally she can record that R1 has been successfully implemented in the IT system.

Step 7: Restore previous. The BA has also to take note that at the end of the year she has to check out her agenda, to ensure that rule R1 and related changes are canceled and the company processes are restored to the previous state. At this point, the IT engineer will have to modify the program code again and the BA will have to cancel all changes relating to R1 in the workflows. Likely, the code cannot simply be restored to the version preceding the changes done to accommodate rule R1, as other changes, due to other variations, may have been included in the version that is operative at the end of the year.

3 StPowla

As we said above, StPowla integrates three ingredients: workflows, policies, and SOA. Workflows are used to define the business process core as the composition of building blocks called *tasks*, à la BPMN. Each task performs a meaningful step in the business, whose purpose is well understood at an abstract level by the stakeholders. That is, a task is understood as to its effect in the business, regardless of the many details that need to be fixed in its actual enactment.

Policies are used to express finer details of the business process, by defining functional and non-functional requirements of task *executions*. The added value is that policies can be updated dynamically, to adapt the core workflow to the changing needs of the various stakeholders.

3.1 Concepts.

Tasks are the units where BPM, SOA and policies converge, and where adaptation occurs: we revisit the intuitive notion of task to offer a novel combination of services and policies. To adapt a workflow to their needs, the stakeholders can influence the behaviour of the tasks by using policies. The UML2 class diagram in Figure 2 presents the related concepts. A `Task` has a `TaskType`, whose name and `description` convey its purpose: each `TaskType` identifies precise functional requirements in the domain. The `name` of the task is useful to distinguish different occurrences of the same task type in the same workflow.
`TaskType` specializes `Type`: besides denoting a set of values and operations, as usual, it specifies a number of task `Attribute`s and `Dimension`s that play an essential role in policy definition. Indeed, as already mentioned, a task is carried out by a service in StPowla: each dimension in the task type specifies the range of variability of the service level, with respect to one kind of resource. The stakeholders can specify the requested levels the service must `comply` to along each dimension, by writing their policies. To ensure that ranges can be defined, dimensions are restricted to be totally ordered.

For instance, `Predelivery` the type of the task of interest in our scenario (VoIP.3), is defined in Table 1: the task has an *offer* as input and returns a *contract*, and its adaptability space is defined along three SL dimensions, each

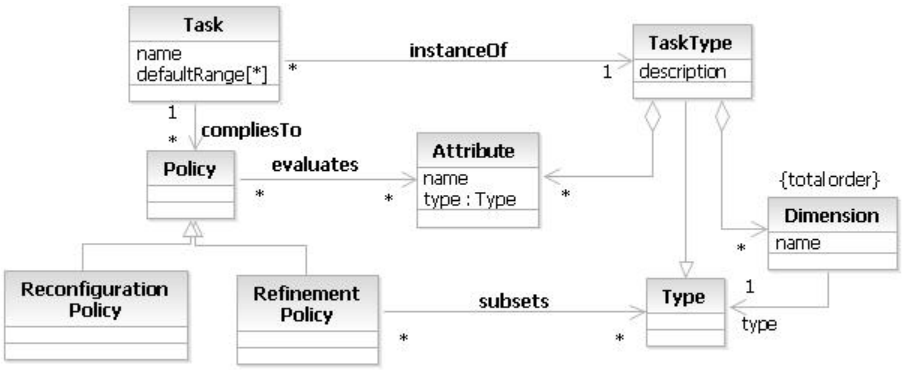

Figure 2: StPowla Concepts

Name:	PreDelivery	**Description:**	Testing & Contract Issuing
Input:	serviceOffer: ServiceOffer	**Output:**	contract: Contract
SL Dimensions and default			
LegalAssistance = [normal], Signature = [normal], TestingEffort = [medium]			

Table 1: The type of task VoIP.3

with the specified `defaultRange`. A typical default value is the entire domain, that is, no constraint at all. The dimensions are defined by the designer in a table, whose relevant excerpt is shown in Table 2. We assume a standard OO format for the definition of the data types, like ServiceOffer and Contract.

The requested service levels may also be specified taking into account the state of execution: task attributes hold properties of individual tasks, workflow attributes of the workflow, and global attributes maintain properties like time, etc. StPowla users can capture the state of the execution `evaluating` them in policies.

Finally, attributes are specified at design–time and bound at run–time, e.g. on task entry, as a function of the task inputs, and of the other attributes.

A task may have associated `Policies`, which come in two flavours: those that adapt the workflow by constraining the task behaviour along its SL dimensions (`RefinementPolicies`), and those that modify the workflow structure, adding

Name	**Values**
...	...
LegalAssistance	none, normal
Signature	normal, double
TestingEffort	none, low, medium, high
...	...

Table 2: VoIP SL dimension

and/or deleting tasks (`ReconfigurationPolicies`). The latter are discussed in [5]; here we will concentrate on refinement policies, and call them simply policies. Before looking at policies, a quick overview of the run–time support is useful.

3.2 Run–time support.

The merging of processes and policies occurs at execution time; that is, each instance of the workflow might be executed differently depending on the environment and the applicable policies. In this section we provide insight into the run-time environment and how this ensures that policies are enforced; however the fine technical details are beyond the scope of this paper. The runtime system consists of a two layer architecture. Figure 3 presents an overview, where the topmost layer is a user interface layer that allows for formulation of policies and upload of the same to the policy server. The bottom layer consists of the workflow execution environment. The top layer is referred to as policy server layer; it is here that policies are stored, evaluated and actions are initiated. The actions are executed by the workflow engine. We will now describe the relevant parts in more detail.

As we say next, we expect users to use tables as a natural policy specification mechanism. The process of deploying a policy means that the corresponding table is sent to the policy server which has a built-in compiler to convert it into its own representation.

The policy server is core to the whole architecture: it provides interfaces for users to deploy new policies and it interacts with the workflow engine to enforce the policies at execution time. The workflow engine temporarily halts the execution when a new task is entered and passes the trigger (task_entry) to the policy server together with some environment information, i.e. the data for the attributes and the evaluation of the SL constraints. The policy server evaluates its policies and determines the actions to take. Usually these will establish the requirements for the service that should be executed to complete the task. At this point the policy server invokes a service–lookup engine to obtain a suitable endpoint that it passes to the workflow engine, which finally resumes the execution of the refined workflow. That is, StPowla relies on *invocation time* service binding, in the terminology of [17].

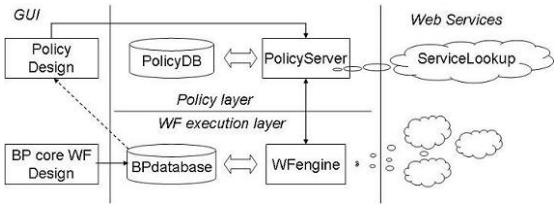

Figure 3: StPowla run–time architecture

3.3 Policy representation.

Policies are specified in tabular form, whose structure is exemplified in Table 4. From top to bottom: the first rows associates the table with the task (VoIP.3) and the trigger (`TaskEntry`) it applies to (the task type permits local checks). The rest of the table is divided in two: on the right side the constraints, on the left side the conditions for their applicability when the policy has been triggered. On the right side, the first row specifies how the policy affects the SL dimensions: each of the rows below defines one possible set of constraints. The default value is taken if the entry is empty.

On the left side, we have a tabular representation of the decision tree to select the appropriate set of constraints: the top cell in each column shows the type of the discriminating values; below it is the expression to `evaluate` the discriminator on task entry. The next rows are the actual encoding of the tree: if the leftmost discriminating value falls in the top range, and the other discriminating value also falls in the top range of its column, then the top row of constraints applies; if, on the contrary, the other value falls in the second range of its column, then the second row applies; and so on. This encoding extends obviously to any number of discriminators.

Finally, the rule names in the requested service level entries are there for traceability reasons: given the policy definition process outlined in the previous section, they permit to recover the business rules that motivated the business analyst to introduce each constraint.

The details of the language used to express values and ranges in policy tables, largely depend on the implementation, and specifically on the policy engine. In this paper, for sake of space, we skept the issue using only enumeration types and fixed values, respectively.

3.4 Implementation issues.

StPowla is intended to be independent of the workflow notation: we have been using the SENSORIA UML4SOA profile [13] to accommodate useful workflow operators.

StPowla is also intended to be independent of the policy language. In previous work we envisaged that the Business Analyst (BA) would express policies in a well established policy language, namely Appel [19], whose run–time has been successfully used for telecommunication services [23]. The approach has

Policies for VoIP.3 : PreDelivery **when** taskEntry				
Discriminators		**Requested SLs**		
CustomerType	OrderRange	LegalAssistance	SignatureType	TestingEffort
thisTask.customerType	thisTask.orderRange			
individual	small	Rx : none		
	medium, large	Rx : none	Ry : double	
business	small			
	medium, large		Ry : double	

Figure 4: An example of policy table

several advantages, but a major drawback, in that often the BA is not computer literate enough to be confident with the language. The tables introduced here can be easily compiled in the XML format of APPEL.

The lookup service developed in the inContext project [22] is a natural candidate for service search, as one of the authors is involved in the project. However, any other service lookup that can deal with SL requests would do.

We are now ready to review the Telco scenario.

4 Implementing R1 in StPowla

We now follow the STPOWLA BA in addressing rule `R1`, assuming the previous design of the business process. We keep the same numbering as in Section 2, to facilitate the comparison.

Step 1: Search. The BA searches the STPOWLA Business Process repository for tasks that consume resources for testing, i.e. that feature the `TestingEffort` dimension.

Step 2: Found. The search returns a table with pointers to the relevant task definitions, in particular to VoIP.3, as per Table 1.

Step 3: Analyze/change. The BA analyzes the current policy table associated to VoIP.3 (Figure 4), to check the policy for small orders from individual customers. Therefore, the BA has to consider the task policy table, The table shows that the PreDelivery discriminators already include `CustomerType` and `OrderRange`, and that testing is always executed by default at level `medium`. Therefore, to implement R1 it is enough that the BA changes, in the Testing-Effort column of the policy table, the uppermost empty entry to 'R1 : none'. Indeed, this entry relates to individual customers and small orders.

Step 4: Continue. The search in step 2 returns all the task featuring the `TestingEffort` dimension. Therefore looping through all the tasks is much easier. Once all the relevant entries are updated as shown in step 3, the tables can be re–deployed in the policy server: R1 is implemented and the Technical Department is free to concentrate on the critical order, till the end of the year.

Step 5: Record. Now, the BA has to record the implemented business rule R1 in his records; he probably also wishes to record the changes that have been applied. Again a support tool is envisioned to help here.

Step 6: Hand to IT. There is no longer need to hand a change request over to the IT Department; as previously noted, the implementation is now automatic.

Step 7: Restore previous. Potentially, the workflow engine could take into account the new business rule only within its period of validity (until the end of the year), by means of 'temporal discriminators'. The flexibility introduced by the business rules substitutes in some cases the software version management procedures.

5 Discussion

5.1 Design issues.

The process to manage adaptation that underlies StPowla involves at least two roles. One is the BA, who is in charge of business process modelling, the other is a catch-all role, Stakeholder (SH). The latter can be played by any person involved with the business process, usually with only cursory knowledge of the business process, but with an interests in its performance and/or outcomes. Because of these interests the SH will from time to time express new requirements that, once approved by the appropriate level of management, are passed to the BA, in the form of business rules, in natural language. The BA will then translate the rule into appropriate policy tables for the appropriate tasks of the workflows. Often, as it is usual in software development tasks, the BA will need to report back to the proposing SH to revise the requirement in the light of his knowledge of the workflow. Anyhow, the table prepared by the BA will be compiled into the policy language and deployed in the policy server, from where it will influence the behavior of the processes as they are executed.

Attributes, and their types, are identified and introduced at different points, during the development of the business model: a few types are *predefined*, i.e. they come with StPowla and are applicable to any task. Most of the attributes types are identified in the construction of the *domain model*, while others may be added for the needs of specific tasks.

As it has been exemplified in the scenario above, the identification of attributes and SL dimensions is a key design activity in StPowla. Indeed, not only the business process flexibility depends essentially on the BA's foresight in this respect, but these choices have also an impact on the service implementations. In fact, any relevant combination of SL values may lead to the need of a specific service. Product lines techniques might provide useful guidelines in this respect.

5.2 StPowla benefits.

From the discussion on deploying `R1`, we can identify the following ones:

- The search in the business process space is IT supported, so that results are easily obtained and more precise (steps 2.1 and 2.2);
- The analysis of change impacts and business rule implementation are simpler, especially if the business process structure has been designed to distinguish the items (conditions of applicability and SL constraints) addressed by the new rule, as shown above (step 2.3). Then, the changes are almost immediately implemented in the IT system;
- Some steps can be automated (2.6 and 2.7). Software configuration management is also reduced.

In general, StPowla potentially improves the company efficiency, by lowering the BA and IT labour costs, as well as its effectiveness, thanks to the shorter time–to–market, and to less errors in BP design and programming.

When several policies are composed or applied simultaneously they might contradict each other. A logic–based approach to detect policy *conflicts*, applicable to StPowla, is presented in [14].

6 Related Work

Much work has been published in the area of business process specifications, ranging from natural English to structured languages used for expressing processes. BPEL [11] is considered the de–facto standard for SOA-based business processes, despite its initial purpose as a service composition language. More traditional workflow languages such as YAWL [24] may be considered better in terms of describing processes since they abstract away composition details that would be included in those solutions previously discussed.
Policies are descriptive and essentially provide information that is used to adapt the behaviour of a system. Most work deals with declarative policies. Notable examples are the formalisms for SLA, i.e. to specify client requirements and service guarantees, and to *sign* an agreement between them [7, 6].

Ideas of introducing flexibility into workflows have been presented by Reichert and Dadam [18] and in the Woklet system by Adams et al [3]. The formers discuss a framework for dynamic process change, but do not include support for changes to the workflow in progress. The latter is based on an extensible repertoire of sub–processes aligned to each task, one of which is chosen at runtime. The difference here is that our adaptation results on changing the Service Levels. Though less general, our approach provides a guidance during the design phase.

Possibly *AgentWork* [15], where rules can be used to drop or add individual tasks to workflows, is close to our reconfiguration policies [9, 10]. However, there is no notion of tasks being linked to services in this work, and the policies are concerned with task replacement rather than task implementation or service selection.

A complementary approach, also combining rules and workflows, is taken in [21]. Here, the objective is compliance control, with respect to regulatory, standard–driven and business–driven requirements. They introduce a control process under the responsibility of a compliance expert, and a computationally efficient non-monotonic deontic logic of violations to model the constraints.

Among the various types of software tools available in the marketplace for BPM support, several business rules management tools (BR tools) became available in recent years. Some vendors offer solutions for doing discovery and analysis of business rules, other offer repositories to document the business rules and still others offer tools to automate business rules, such as business rule engines or code generators from business rules [20]. Among the most complete and promising solutions are Blaze Advisor [1] and JRules [2]. Recently BR tools have been including SOA integration features, such as deploying rule services as part of an SOA [16].

7 Conclusions

StPowla introduces a novel combination of policies and workflows that adds to each of the concepts being used on their own, so allowing the designer to capture the essential requirements of a business process using a workflow notation and at the same time permits for the variability to be expressed in a descriptive way by policies. Additionally, StPowla creates a clear link between this enhanced workflow mechanism and services: tasks are being executed by services, and StPowla allows the BA to design functional and SL requirements that together characterize the guarantees the chosen service has to provide.

Acknowledgments.

All the authors are partially supported by the EU project SENSORIA IST-2005-16004.

References

[1] http://www.fairisaac.com/fic/en/product-service/product-index/blaze-advisor/. Last visited: May 2008.

[2] http://www.ilog.com/products/businessrules/index.cfm. Last visited: May 2008.

[3] M. Adams, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Worklets: A service-oriented implementation of dynamic flexibility in workflows. In R. Meersman and Z. Tari, editors, *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE, OTM Confederated International Conferences*, volume 4275 of *LNCS*, pages 291–308. Springer, 2006.

[4] L. Baresi, E. Di Nitto, and C. Ghezzi. Towards Open-World Software. *IEEE Computer*, 39:36–43, October 2006.

[5] L. Bocchi, S. Gorton, and S. Reiff-Marganiec. Engineering Service Oriented Applications: From StPowla Processes to SRML Models. 2008. To Appear in FASE08.

[6] M.G. Buscemi, L. Ferrari, C. Moiso, and U. Montanari. Constraint-based policy negotiation and enforcement for telco services. In *Proc. 1st IEEE & IFIP Theoretical Aspects of Software Engineering Conference (TASE 2007)*. IEEE Computer Society, 2007.

[7] M.G. Buscemi and U. Montanari. Cc-pi: A constraint-based language for specifying service level agreements. In R. De Nicola, editor, *Programming Languages and Systems (ESOP 2007)*, pages 18–32, 2007.

[8] S. Gorton, C. Montangero, S. Reiff-Marganiec, and L. Semini. StPowla: SOA, Policies and Workflows. 2007. To Appear in Proceedings of WESOA 2007; LNCS.

[9] S. Gorton and S. Reiff-Marganiec. Policy support for business-oriented web service management. In *Proceedings of the Fourth Latin American Web Congress (LA-WEB'06)*, pages 199–202, Washington, DC, USA, 2006. IEEE Computer Society.

[10] S. Gorton and S. Reiff-Marganiec. Towards a task-oriented, policy-driven business requirements specification for web services. In S. Dustdar, J.L. Fiadeiro, and A.P. Sheth, editors, *Business Process Management*, volume 4102 of *LNCS*, pages 465–470. Springer, 2006.

[11] D. Jordan and J. Evdemon et al. Web services business process execution language version 2.0. W3C, Aug 2006. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.pdf.

[12] F. Kamoun. A roadmap towards the convergence of business process management and service oriented architecture. *Ubiquity*, 8(14), 2007. ACM Press.

[13] N. Koch, P. Mayer, R. Heckel, L. Gonczy, and C. Montangero. UML for service-oriented systems, SENSORIA EU-IST 016004 Deliverable D1.4.a. http://www.pst.ifi.lmu.de/projekte/Sensoria/del_24/D1.4.a.pdf, 2007.

[14] C. Montangero, S. Reiff-Marganiec, and L. Semini. Logic-based detection of conflicts in APPEL policies. In F. Arbab and M. Sirjani, editors, *Int. Symp. on Fundamentals of Software Engineering, FSEN 2007, Tehran, Iran*, volume 4767 of *LNCS*, pages 257–271. Springer, 2007.

[15] R. Müller, U. Greiner, and E. Rahm. Agent work: a workflow system supporting rule-based workflow adaptation. *Data Knowl. Eng.*, 51(2):223–256, 2004.

[16] S. Núñez. ILOG JRules 6.5 brings rules to SOA. InfoWorld: Product Guide: ILOG JRules 2007 : Review, 2007.

[17] C. Pautasso and G. Alonso. Flexible binding for reusable composition of web services. In *Software Composition*, volume 3628 of *LNCS*, pages 151–166. Springer, 2005.

[18] M. Reichert and Peter Dadam. ADEPT flex -supporting dynamic changes of workflows without losing control. *J. Intell. Inf. Syst.*, 10(2):93–129, 1998.

[19] S. Reiff-Marganiec, K.J. Turner, and L. Blair. Appel: The accent project policy environment/language. Technical Report TR-161, University of Stirling, Dec 2005.

[20] G. Steinke and C. Nickolette. Business rules as the basis of an organizations information systems. *Industrial Management & Data Systems*, 103(1):52–63, 2003.

[21] S.W.Sadiq, G.Governatori, and K.Namiri. Modeling control objectives for business process compliance. In *BPM2007*, volume 4714 of *LNCS*, pages 149–164. Springer, 2007.

[22] H.-L. Truong, S. Dustdar, D. Baggio, C. Dorn, G. Giuliani, R. Gombotz, Y. Hong, P. Kendal, C. Melchiorre, S. Moretzky, S. Peray, A. Polleres, S. Reiff-Marganiec, D. Schall, S. Stringa, M. Tilly, and H.Q. Yu. incontext: a pervasive and collaborative working environment for emerging team forms. In *Proc. 2008 International Symposium on Applications and the Internet*, 2008. To Appear.

[23] K. J. Turner, S. Reiff-Marganiec, L. Blair, J. Pang, T. Gray, P. Perry, and J. Ireland. Policy support for call control. *Computer Standards and Interfaces*, 28(6):635–649, 2006.

[24] W. M. P. van der Aalst and A. H. M. ter Hofstede. YAWL: yet another workflow language. *Inf. Syst.*, 30(4):245–275, 2005.

[25] M. Wirsing, G. Carizzon, S. Gilmore, L. Gönczy, N. Koch, P. Mayer, and C. Palasciano. A systematic approach to developing service-oriented systems. Technical report, EU IP–IST 016004 SENSORIA, 2007. http://www.sensoria-ist.eu/files/sensoria_whitepaper_20080303.pdf.